REMARKS

Claims 1-40 were pending in the application. Claims 1-2, 17-18, 27-28, and 32-33 have been amended. Claim 40 has been cancelled. Claims 41-47 have been added. Accordingly, claims 1-39 and 41-47 are now pending in the application.

Support for new claim 41 may be found, for example, on page 11, lines 25-26 and page 14, lines 16-18 of the disclosure. Support for claims 42-43 may be found, for example, on page 19, lines 2-8. Support for claim 44 may be found, for example, on page 10, lines 4-6 and page 15, lines 21-24. Support for claim 45 may be found, for example, on page 10, lines 4-13 and on page 14, line 13 - page 15, line 12. Support for claim 46 may be found, for example, on page 10, lines 4-13 and on page 14, line 13 - page 15, line 12. Support for claim 47 may be found, for example, in the original claim 1, on page 14, line 13 - page 15, line 12, and on page 12, lines 23-28.

35 U.S.C. § 102 Rejection

Claims 1-9, 11-25, 27-38, and 40 were rejected under 35 U.S.C. 102(e) as being anticipated by Archibald, Jr. et al. (U.S. Publication # 20020169995A1). Applicant respectfully traverses this rejection.

Archibald teaches, "performing data consistency checks on user data stored in a data storage subsystem." (Archibald, Page 1, Paragraph 0001) More specifically, Archibald teaches,

> "In the data checking portion, first, sectors (data sectors and parity sectors) of a data stripe are read. This reading may involve reading all data and parity sectors depending upon the granularity permitted (or desired) for regions of the data and parity. In one embodiment, all data and parity sectors are read. For each sector stripe, beginning with a first selected sector stripe, a data check code sub-sector stripe (DCCss) is calculated or otherwise generated for the selected sector stripe using the DCCds metadata for each data sector. Next, a comparison is made between the calculated DCCss and the previously stored DCCss to determine if they are the same. Recall that the calculated DCCss is the data check code calculated, computed, or otherwise generated based on the stored DCCds. The

DCC metadata object stored in the sector header for each parity sector is an encoded representation of the DCCds stored in the sector header for each data sector in that sector stripe and the DCCps stored in the sector header of the parity sector in that sector stripe. The stored DCCss was written when the data and parity were originally stored. The calculated DCCss will be the same or equal to the stored DCCss when there has been no data corruption that would introduce inconsistencies. The DCCss generation and comparison operations are performed for each sector stripe, preferably in parallel or otherwise concurrently or with some overlap, and sequential processing for each sector stripe (such as might otherwise be inferred from the flow-chart diagrams) is not required. If no inconsistencies are detected across each sector stripe in the stripe, then the data check procedure is finished for that stripe." (Archibald, Page 4, Paragraph 0035) (Emphasis added)

"If any inconsistencies are detected, the data inconsistency reporting and correcting sub-procedure is performed. This reporting and correction procedure is first briefly highlighted and then described in greater detail. **The data stored in the sector body is used to generate**: (i) a calculated DCCds for each data sector in the sector stripe, (ii) a calculated DCCps for the parity sector in the sector stripe, and (iii) a calculated DCCss for the sector stripe based on the newly calculated DCCds and DCCps. The newly calculated DCCds and stored DCCds for each data sector in the sector stripe are compared, as is the newly calculated DCCps compared to the stored DCCps for the parity sector. If an inconsistency is detected between a calculated DCCds and a stored DCCds for a data sector and auto correction for the user data in enabled, the user sector body of data for the affected sector is regenerated using **the remaining good data** and **parity sector**. Likewise, if an inconsistency is detected between a calculated DCCps and a stored DCCps, the parity data for parity sector is regenerated using **the good host data stored in the stripe sector bodies**." (Archibald, Page 4, Paragraph 0036) (Emphasis added)

Applicant respectfully submits that Archibald fails to teach or suggest "said storage controller is further configured to initialize a given stripe in response to receiving **a write request** to write a **new** data block at **a particular location** of said given stripe and detecting a mismatch in block verification information associated with an existing data block **at the particular location** of said given stripe to be updated, wherein said storage controller is configured to initialize said given stripe by generating a corresponding redundancy data block for said given stripe **based on** at least the **new data block to be written** to said given stripe" as recited in claim 1. As noted above, while Archibald teaches that if an error is detected in a data block, the data block having the error is regenerated using the remaining good data blocks and

the parity block (i.e., existing data) of the stripe, Archibald does not teach initializing a given stripe in response to receiving a write request and detecting a mismatch in the block verification information of an existing data block of the given stripe by generating an updated parity block based on at least a new data block to be written (i.e., new data) to the given stripe. Also, while Archibald teaches that if an error is detected in a parity block, the parity block having the error is regenerated using the good data blocks stored (i.e., existing data) in the stripe, Archibald does not teach initializing a given stripe in response to receiving a write request and detecting a mismatch in the block verification information of an existing data block of the given stripe by generating an updated parity block based on at least a new data block to be written (i.e., new data) to the given stripe.

In accordance, independent claim 1 is believed to patentably distinguish over Archibald. Claims 2-9 and 11-16 depend on claim 1 and are therefore believed to patentably distinguish over Archibald for at least the reasons given above.

Likewise, independent claims 27 and 32 recite features similar to those highlighted above with regard to independent claim 1 and are therefore believed to patentably distinguish over Archibald for at least the reasons given above. Claims 29-31 and claims 34-38 depend on claim 27 and claim 32, respectively, and are therefore believed to patentably distinguish over Archibald for the same reasons.

In addition, Applicant respectfully submits that Archibald fails to teach or suggest "wherein said block verification information associated with a particular data block includes an address associated with said particular data block" as recited in claim 8. Also, Applicant submits that Archibald fails to teach or suggest "wherein said address is a logical block address for said particular block" as recited in claim 9. The Examiner contends that page 3, paragraph [0028] of Archibald teaches the features recited in claims 8 and 9. Archibald teaches on page 3, paragraph [0028], "A particular data stripe 200 is addressed by its logical volume and logical block number." However, Archibald fails to teach or suggest block verification information of a data block includes an address corresponding to the data block. Thus, while Archibald teaches addressing a data stripe

using a logical volume and a logical block number, Archibald does not disclose or suggest using such addressing identifiers as block verification information that is used by a storage controller to detect a mismatch for initialization purposes, as recited in claims 1, 8, and 9. In accordance, claims 8 and 9 are believed to patentably distinguish over Archibald.

Furthermore, Applicant respectfully submits that Archibald fails to teach or suggest "said storage controller is further configured to initialize a given stripe in response to receiving **a write request** to write a new data block at **a particular location** of said given stripe and detecting a mismatch in block verification information in each of **at least two** existing data blocks of said given stripe, wherein one of the two existing data blocks is **at the particular location** of said given stripe to be updated, wherein said storage controller is configured to initialize said given stripe by generating a corresponding redundancy data block for said given stripe **based on** at least **the new data block to be written** to said given stripe" as recited in claim 17. As noted above, Archibald teaches that if an inconsistency is detected between a calculated DCCds and a stored DCCds, the corresponding data block is regenerated using the remaining good data blocks and the parity block of the stripe. Also, Archibald teaches that if an inconsistency is detected between a calculated DCCps and a stored DCCps, the corresponding parity block is regenerated using the good data blocks stored in the stripe. However, Archibald fails to teach or suggest "generating a corresponding redundancy data block for said given stripe based on at least **the new** data block **to be written** to said given stripe."

Additionally, Archibald teaches "If the error is with one of the DCCds or the DCCps, then the number of errors becomes important. If there is one error with the DCCps and DCCds, the error may be correctable, however if there is more than one error (326) with the DCCps and DCCds, an uncorrectable error is optionally but preferably logged." (Archibald, Page 5, Paragraph 0043) (Emphasis added) Thus, Archibald fails to teach or suggest "initialize a given stripe in response to receiving a write request to write a new data block at a particular location of said given stripe and detecting a mismatch in block verification information in each of at least **two** existing data blocks of said given stripe."

In accordance, independent claim 17 is believed to patentably distinguish over Archibald. Claims 19-25 depend on claim 17 and are therefore believed to patentably distinguish over Archibald for at least the reasons given in the above paragraphs discussing claim 17.

Also, Claim 40 has been cancelled, therefore the rejection of claim 40 is now believed moot.

Additionally, Applicant respectfully requests examination of added claims 41-47.

Claims 41-46 depend on claim 1 and are therefore believed to patentably distinguish over Archibald for at least the reasons given in the above paragraphs discussing claim 1.

Applicant respectfully submits that Archibald fails to teach or suggest "receiving a **write request** to write a new data block at **a particular location** of a given stripe; in response to receiving the write request, **reading an existing data block at the particular location** of said given stripe to be updated; and initializing said given stripe in response to detecting a mismatch in block verification information associated with the existing data block, wherein said initializing said given stripe comprises generating a corresponding redundancy data block for said given stripe based on at least the **new** data block **to be written** to said given stripe" as recited in claim 47. In accordance, independent claim 47 is believed to patentably distinguish over Archibald.

35 U.S.C. § 103 Rejection

Claims 10, 26, and 39 were rejected under 35 U.S.C. §103(a) as being unpatentable over Archibald in view of IBM. Applicant respectfully traverses the rejection. Applicant believes claim 10, claim 26, and claim 39 are patentably distinguishable as dependent on claim 1, claim 17, and claim 32, respectively, which are believed to patentably distinguish over Archibald for at least the above stated reasons.

CONCLUSION

In light of the foregoing amendments and remarks, Applicant submits that all pending claims are now in condition for allowance, and an early notice to that effect is earnestly solicited. If a phone interview would speed allowance of any pending claims, such is requested at the Examiner's convenience.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-77800/BNK.

Respectfully submitted,

B. Noël Kivlin
Reg. No. 33,929
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, Texas 78767-0398
Phone: (512) 853-8800
Date: 11-5-04